Formalizing the Cardiac Pacemaker Resynchronization Therapy

Neeraj Kumar Singh(✉), Mark Lawford, Thomas S.E. Maibaum, and Alan Wassyng

McMaster Centre for Software Certification, McMaster University, Hamilton, ON, Canada
{singhn10,lawford,wassyng}@mcmaster.ca, tom@maibaum.org

Abstract. For many years, formal methods have been used to design and develop critical systems in order to guarantee safety and security and the correctness of desired behaviours, through formal verification and validation techniques and tools. The development of high confidence medical devices such as the cardiac pacemaker, is one of the grand challenges in the area of verified software that need formal reasoning and proof-based development. This paper presents an example of how we used previous experience in developing a cardiac pacemaker using Event-B, to build an incremental proof-based development of a new pacemaker that uses Cardiac Resynchronization Therapy (CRT), also known as biventricular pacing or multisite pacing. In this work, we formalized the required behaviours of CRT including timing constraints and safety properties. We formalized the system using Event-B, and made use of the included Rodin tools to check the internal consistency with respect to safety properties, invariants and events. The system behaviours of the proven model were validated through the use of the ProB model checker.

Keywords: Pacemaker resynchronization therapy · Event-B · Refinement · Formal methods · Verification · Validation

1 Introduction

Patient safety is a global challenge that requires practical knowledge and technical skills in clinical assessments, embedded systems, and software engineering including human factors and systems engineering (HFE). Many incidents related to patient safety are due to lack of attention to HFE in the design and implementation of technologies, processes, and usability. The main objective of HFE is to improve system performance including patient safety and technology acceptance [1]. The USA Food and Drug Administration (FDA) has reported several recalls in which pacemaker and implantable cardioverter-defibrillator (ICD) failures are responsible for a large number of serious illnesses and deaths. According

Partially supported by: The Ontario Research Fund, and the National Science and Engineering Research Council of Canada.

© Springer International Publishing Switzerland 2015
V.G. Duffy (Ed.): DHM 2015, Part II, LNCS 9185, pp. 374–386, 2015.
DOI: 10.1007/978-3-319-21070-4_38

to the FDA, 17,323 devices (8834 pacemakers and 8489 ICDs) were explanted during 1990-2002, and 61 deaths (30 pacemaker patients, 31 ICD patients) were found to be due to device malfunction. The FDA found that these deaths and other adverse-events were caused by product design and engineering flaws including firmware problems [2]. Critical systems such as the pacemaker need to be better designed to provide the required level of safety and dependability.

Software plays a vital role in developing and controlling medical devices. In order to make sure that medical devices are safe, secure and reliable, regulatory agencies like the FDA, require evidence based criteria to approve these devices. Over the past twenty years, formal techniques have shown some promising results in the health care domain through identifying abnormal behaviours or possible errors by applying mathematical reasoning.

In 2003 Tony Hoare suggested a verification grand challenge for computing research. In similar vein, a real ten year-old, sanitized pacemaker specification [3] was used by the Software Quality Research Laboratory at McMaster University, to issue the *PACEMAKER Challenge* to the software engineering community. The challenge is characterized by system aspects emphasizing the development and certification of dependable and safe pacemakers, and is now managed by the McMaster Centre for Software Certification. Many researchers have worked and are working on the PACEMAKER Challenge [4], but most of them are focusing on 1-and 2-electrode pacemakers. In this paper, we demonstrate the results of our new work on the formalization of the system requirements for the *Cardiac Resynchronization Therapy (CRT)* pacemaker, or multi-site pacing pacemaker that can be used to help certify the CRT pacemaker. Our main objectives and contributions are as follows:

- To build on experience in the PACEMAKER Challenge in using Event-B to develop a CRT pacemaker;
- To further develop principles of how to use refinement in Event-B effectively, to model the required behaviour of a medical device;
- To formalize and analyze the behavioural requirements for the CRT pacemaker;
- To define a list of safety properties;
- To verify and validate the system requirements of the CRT pacemaker;
- To demonstrate how we can help to meet FDA requirements for certifying the CRT pacemaker using formal methods;

To formalize the CRT device we selected the Event-B modelling language [5], so that we could use previous experience to guide the refinement steps, and also because of the formal tools that were then available to us. Event-B supports traditional refinement in which each refinement step adds detail to existing functionality. It also supports (horizontal) refinement in which the steps add new functionality in the solution of the problem. This (horizontal) refinement allows us to develop an incremental approach to building these safety-critical medical devices. The incremental steps to use are not always obvious, and this is where previous experience plays an important role.

This incremental development preserves the required behaviour of the system in the abstract model as well as in the correctly refined models. The Event-B language is supported by the Rodin platform, which provides a rich set of provers and other supporting tools for developing the specifications. The Rodin platform helps us guarantee the preservation of safety properties. We use the ProB model checker tool [6] to analyze and validate the developed models of the CRT pacemaker.

The remainder of this paper is organized as follows. Section 2 presents preliminary information about the CRT pacemaker. The CRT pacemaker control requirements are presented in Sect. 3. Section 4 explores an incremental proof-based formal development of the CRT pacemaker. Brief discussion is provided in Sect. 5. Section 6 presents related work. Section 7 concludes the paper and presents future work.

2 Preliminaries

The cardiac pacemaker is an electronic device equipped with a microprocessor, and is designed to maintain regular heart beats. The pacemaker generally serves two main functions: *sensing* and *pacing*. Sensors are used to sense the intrinsic activities of the heart chambers, and when appropriate, actuators are used to deliver a short intensive electrical pulse into the heart chambers.

A CRT or multi-site pacing device is an advanced pacemaker that is designed to treat a specific form of heart failure – poor synchronization of the two lower heart chambers. This device sends a very low power electrical impulse to both lower chambers of the heart to help them beat together synchronously. As we said, it is an advanced pacemaker, so it carries all the functional behaviours of a simple pacemaker including these more advanced features to synchronize the lower chambers of the heart. The basic elements of a pacemaker system [7] are:

- **Leads:** A set of insulated flexible wires, used to transmit electrical impulses between the microprocessor and the heart for sensing and pacing purposes.
- **The CRT Generator:** The main unit consisting of battery and microprocessor to control the entire functionality of the system.
- **Device Controller-Monitor (DCM):** An external device that interacts with the implanted CRT using wireless for configuration and for setting new parameters.
- **Accelerometer:** A motion sensor to measure body motion to allow modulated pacing and sensing.

2.1 Event-B

Event-B [5,8] is a modelling language that enables us to formalize a system through stepwise refinement. We can thus build a complex system incrementally by introducing more detail in each refinement step, where each step is verified by generated proof obligations with respect to an abstract model. This refinement process finally culminates in a concrete implementation. The basic system

modelling components are *context* and *machine*. The *context* describes static behaviour, while the *machine* describes the dynamic behaviour of the system using events. At each refinement step, the events can be refined by: (1) keeping the event as it is; (2) splitting an event into several events; or (3) refining by introducing another event to maintain state variables. Importantly, new refinement levels allow the introduction of new events. Refinement in Event-B is an essential component of the methodology. It is important to realize that many of the refinement steps in Event-B represent a decomposition of the strategy, while other refinement steps (more traditional refinement) are a decomposition of the system itself. A set of tools, the *Rodin* [8] *tools*, support model development, proof obligation generation for refinement steps and state predicates, and the discharging of generated proof obligations using automated theorem provers. Due to page limitations, we have not presented a detailed introduction to Event-B. There are numerous publications and books available for an introduction to Event-B and related refinement strategies [5,8].

3 CRT Pacemaker Control Requirements

The focus of this work is the formalization of biventricular sensing with biventricular pacing (BiSP) of CRT devices. There are various situations, where a CRT device can be used to control the heart rhythm. However, we are interested in formalizing the most complex mode (BiSP) since it also covers the other less complex modes. BiSP allows pacing and sensing in the right atrium, right ventricle, and left ventricle chambers (see Fig. 1) and this section only describes the control requirements of BiSP.

Biventricular pacing coordinates the left ventricle (LV) and right ventricle (RV), and intra-ventricular regional wall contractions. It also synchronizes pacing with the sinus rhythm sensed in the right atrium. There are various intrinsic events for LV and RV, like sensing and pacing occurrences, which can reset pacing intervals that produce several variations of the atrioventricular interval (AVI).

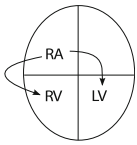

Fig. 1. Biventricular pacing.

Biventricular pacing controls the heart rate using various combinations of the timing from events in either LV or RV. For instance, the first ventricular sense either from the left or right ventricular chamber can reset the ventriculoatrial interval (VAI) and the heart rate depends on intervals between the first ventricular events in each cycle. However, heart rate intervals can vary due to stimulation in the opposite chambers.

Delays between RV and LV pacing introduce complications in biventricular timings. These timings allow multiple definitions of atrioventricular (AV) and ventriculoatrial (VA) escape intervals. The pacing rate is the sum of the VA and AV escape intervals for dual chamber timing. This definition can be preserved for biventricular timing if the VAI and AVI refer to pacing either the RV for RV-based timing or the LV for LV-based timing. Then the pacing delay can be represented by the RV-LV interval. This interval can be negative, positive or zero as per the occurrence order of the stimulations in both ventricles.

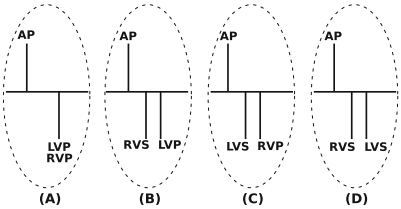

Fig. 2. Possible scenarios of the biventricular sensing and pacing. AS = atrial sensed; AP = atrial paced; LVS = left ventricular sensed; LVP = left ventricular paced; RVS = right ventricular sensed; RVP = right ventricular paced.

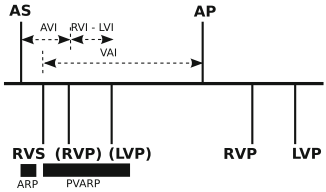

Fig. 3. Biventricular sensing and pacing (BiSP) with a RVS event

Figure 2 depicts the possible scenarios in a sequential order for biventricular sensing and pacing [9]. The possible scenarios are described as follows, assuming normal pacing and sensing activities in the right atrium chamber:

- **Scenario A** shows a situation in which the pacemaker paces in both ventricles after an AV interval in which no intrinsic heart activity is detected.
- **Scenario B** shows a situation in which the pacemaker paces in LV only after an AV interval, while RV pacing is inhibited due to sensing of an intrinsic activity in RV.
- **Scenario C** shows a situation in which the pacemaker paces in RV only after an AV interval, while LV pacing is inhibited due to sensing of an intrinsic activity in LV.
- **Scenario D** shows the case where pacing activities are inhibited in the ventricles due to sensing of intrinsic activities in both LV and RV.

There are various possible scenarios to show biventricular sensing and pacing in order to capture possible behavioural requirements. For example, Fig. 3 presents a scenario for biventricular sensing and pacing, in which an event sense related to the right ventricle resets all the pacing intervals for both the right and left ventricles, so pacing is not allowed in the right ventricle or in the left ventricle following a RVS, and a RVS event resets the timing cycle and starts a new VAI. Other requirements are omitted here.

4 Formal Development of CRT

Abstract Model: An abstract model of the CRT pacemaker specifies only pacing and sensing behaviour of three electrodes for each chamber (RA, RV, LV). In order to start the formalization process, we need to define some static properties of the system using Event-B context. The first context declares an enumerated set *Status* as $partition(Status, \{ON\}, \{OFF\})$. This enumerated set is used to specify the *ON* and *OFF* states of the actuators and sensors of the CRT pacemaker.

The CRT pacemaker delivers a pacing stimulus in the RA, RV, and LV as per the patient needs through sensing the intrinsic activities of the heart. The CRT

is much more complex than the 1- or 2-electrode pacemaker, because the CRT pacing behaviour intelligently maintains the synchronicity between RV and LV. The Event-B model declares a list of variables for defining actuators and sensors. *PM_Actuator_A*, *PM_Actuator_LV*, and *PM_Actuator_RV* are actuators for each chamber, and *PM_Sensor_A*, *PM_Sensor_LV*, and *PM_Sensor_RV* are sensors for each chamber, which are defined as the type of *Status* using invariants.

At this stage the system describes only discrete functional behaviour for changing between two states (ON and OFF) without considering any timing requirements. We introduce twelve new events for specifying the pacing and sensing activities in terms of changing states. These events include a guard related to the current state of actuators and sensors, and the action changes the states of the actuators and sensors. An Event *PM_Pacing_On_A* is used to set *ON* for the right atrium actuator, when the right atrium actuator is *OFF*. Similarly, another event *PM_Sensing_On_A* is used to set *ON* of the right atrium sensor, when the atrium sensor is *OFF*. The other events are formalized in a similar way.

EVENT PM_Pacing_On_A
WHEN
grd1 : $PM\_Actuator\_A = OFF$
THEN
act1 : $PM\_Actuator\_A := ON$
END

EVENT PM_Sensing_On_A
WHEN
grd1 : $PM\_Sensor\_A = OFF$
THEN
act1 : $PM\_Sensor\_A := ON$
END

4.1 First Refinement: Timing Requirements

This refinement introduces the timing requirements by defining a logical clock. A list of constants are defined in a new *context* for specifying the desired timing requirements for controlling the pacing and sensing behaviours. We define four constants *AVI*, *VAI*, *LVI*, and *RVI*. The *AVI* allows a value within a range (50 .. 350). The *VAI* allows a value within a range (350 .. 1200). The *RVI* and *LVI* have similar timing intervals (0 .. 50). An extra axiom specifies that the *RVI* should be greater than or equal to *LVI*. In this study we consider all times to be in milliseconds.

A variable *now* is defined to represent the current clock counter in *inv*1, which progresses by 1 millisecond every clock tick. The next variable *PSRecord* is used to store a time when a pacing or sensing activity occurs using *inv*2. The stored time can be used for deciding future pacing or sensing activity in any chamber. A few variables are defined to synchronize the pacing and sensing activities by capturing the different states of sensors and actuators for the atrium and ventricular chambers in order to define the desired behaviour.

$inv1 : now \in \mathbb{N}$
$inv2 : PSRecord \in \mathbb{N}$
$inv3 : now = 0 \Rightarrow PM\_Sensor\_RV = OFF \wedge PM\_Actuator\_RV = OFF$
$inv4 : now = 0 \Rightarrow PM\_Sensor\_LV = OFF \wedge PM\_Actuator\_LV = OFF$
$inv5 : now = 0 \Rightarrow PM\_Sensor\_A = OFF \wedge PM\_Actuator\_A = OFF$
$inv6 : PM\_Actuator\_RV = ON \Rightarrow now \geq AVI \vee Immd\_Pace\_RV = 1$
$inv7 : PM\_Actuator\_LV = ON \Rightarrow$
$\quad now \geq AVI + (RVI - LVI) \vee Immd\_Pace\_LV = 1 \vee Delay\_Pace\_LV = 1$
$inv8 : PM\_Actuator\_A = ON \Rightarrow$
$\quad now \geq PSRecord + VAI \vee now \geq AVI + VAI$

A list of safety properties can then be introduced using invariants. Invariants (*inv*3, *inv*4, and *inv*5) state that when the current clock counter is zero then all the actuators and sensors are *OFF*. It means, the sensor and actuator of each chamber should be *OFF* at the beginning of the pacing cycle. The next safety property *inv6* states that the actuator of the right ventricle must pace when the AVI is elapsed or an immediate pacing is required. The next safety property *inv7* shows that the actuator of the left ventricle must pace when the total duration of the atrioventricular interval and pacing delay is elapsed, an immediate pacing is required, or a delay pacing is detected in the left ventricle. The last safety property *inv8* states that the actuator of the right atrium must pace when the VAI is elapsed after detecting the last pacing or sensing activity, or the total duration of AVI and VAI is elapsed.

EVENT PM_Pacing_On_A Refines PM_Pacing_On_A
WHEN
grd1 : $PM\_Actuator\_A = OFF$
grd2 : $(now = AVI + VAI \wedge No\_Pace\_LV\_RV = 0 \wedge Delay\_Pace\_LV = 0)$
$\vee$
$(now = PSRecord + VAI \wedge (Delay\_Pace\_LV = 2 \vee Delay\_Pace\_LV = 1 \vee$
$No\_Pace\_LV\_RV = 1 \vee RV\_Delay\_AVI = 1 \vee Immd\_Pace\_RV = 1$
$\vee Immd\_Pace\_LV = 1))$
grd3 : $PM\_Sensor\_A = OFF$
grd4 : $Pace\_A = 0$
THEN
act1 : $PM\_Actuator\_A := ON$
act2 : $Pace\_A := 1$
END

In this refinement, we introduced several new events to specify the desired behaviour of actuators and sensors according to the given timing requirements. The complete formal specification can produce an algorithm for implementation purpose. There are eighteen events in total, in which seventeen events refine the events of the abstract model. For example, the event *PM_Pacing_On_A* refines the abstract event *PM_Pacing_-On_A* by adding new guards and adding a new action.

The pacing and sensing events update a state every millisecond. We model this increment by a new event *tic*, that increments time by 1 ms. The event *tic* progressively increases the current clock counter *now*. The event *tic* has no guard in this refinement, but in further refinements, we introduce a guard to control the pacing and sensing activities within restricted time intervals.

EVENT tic
WHEN
THEN
act1 : $now := now + 1$
END

4.2 Second Refinement: Threshold

An intrinsic activity of a chamber can be sensed by using the inbuilt sensor of an electrode. The pacemaker can deliver stimulation to the heart chamber based on monitored values and according to the selected safety margin. Each chamber of the heart contains a range of standard threshold values that can be pre-specified by a physiologist[1] for comparing with monitored values to detect the intrinsic

[1] Standard threshold constant values of atria and ventricular chambers are different.

activities. A set of constants (STA_THR_A, STA_THR_LV, STA_THR_RV) is defined to hold a range of standard threshold values for each chamber.

The pacemaker sensor starts sensing intrinsic activities during certain intervals to avoid sensing errors. A pacemaker actuator delivers a small intense electric pulse whenever the natural pace is absent and intrinsic activity is not detected by the sensor. A sensor can detect an intrinsic activity when the threshold value of the detected signal is greater than or equal to the standard threshold constant. In this refinement, we introduce the threshold for right atrium, left ventricle and right ventricle. An invariant ($inv1$) is defined for the right atrium, right ventricle and left ventricle to indicate the boolean states for synchronization purpose to maintain the order of sensing activities in the right atrium and both right and left ventricles. Three safety properties are introduced using invariants ($inv2$, $inv3$ and $inv4$) that state the sensor of each chamber is OFF when the detected sensor value is greater than or equal to the standard threshold value and boolean state of the chamber is TRUE.

$$
\begin{aligned}
&\text{inv1} : Thr\_State\_A \in BOOL \land Thr\_State\_LV \in BOOL \land Thr\_State\_RV \in BOOL \\
&\text{inv2} : \forall i \cdot i \in \mathbb{N}_1 \land i \geq STA\_THR\_A \land Thr\_State\_A = TRUE \Rightarrow PM\_Sensor\_A = OFF \\
&\text{inv3} : \forall i \cdot i \in \mathbb{N}_1 \land i \geq STA\_THR\_LV \land Thr\_State\_LV = TRUE \Rightarrow PM\_Sensor\_LV = OFF \\
&\text{inv4} : \forall i \cdot i \in \mathbb{N}_1 \land i \geq STA\_THR\_RV \land Thr\_State\_RV = TRUE \Rightarrow PM\_Sensor\_RV = OFF
\end{aligned}
$$

This refinement step enriches the previously defined events through strengthening guards without introducing any new events. The new added guards are used to acquire the intrinsic activity by comparing the sensed threshold value with a standard threshold value for all chambers (right atrium, right ventricle, and left ventricle).

EVENT PM_Sensing_Off_A Refines PM_Sensing_Off_A
ANY Thr_A
WHERE
- grd1 : $PM\_Sensor\_A = ON$
- grd2 : $PM\_Sensor\_RV = OFF$
- grd3 : $PM\_Actuator\_A = OFF$
- grd4 : $PM\_Actuator\_RV = OFF$
- grd5 : $PM\_Sensor\_LV = OFF$
- grd6 : $PM\_Actuator\_LV = OFF$
- grd7 : $Thr\_A \in \mathbb{N}$
- grd8 : $Thr\_A \geq STA\_THR\_A$

THEN
- act1 : $PM\_Sensor\_A := OFF$
- act2 : $PSRecord := 0$
- act3 : $now := 0$

END

For example, the event $PM\_Sensing\_Off\_A$, refinement of an abstract event, is strengthened by introducing a new variable Thr_A and guards ($grd7$, $grd8$). The guard ($grd7$) defines the type of variable (Thr_A), while the next guard ($grd8$) compares the threshold value with the standard threshold value of the right atrium. The new introduced guards allow to monitor an activity of the right atrium by comparing a sensed value with the selected standard threshold value of the atrial chamber. We have modified several other events in similar fashion to model the desired behaviour of sensors.

4.3 Third Refinement: Refractory and Blanking Periods

This refinement introduces refractory and blanking periods[2] for atrial and ventricular chambers. These blanking and refractory periods are used to suppress device-generated artifacts and unwanted signal artifacts. These periods are designed to

[2] https://www.bostonscientific.com/content/dam/bostonscientific/quality/education-resources/english/ACL_Cross-Chamber_Blanking_20081219.pdf.

promote appropriate sensing of intrinsic activities, and to prevent over sensing activities in another chamber. In this refinement, we define eight constants Atrial Refractory Period (ARP), Right Ventricular Refractory Period (RVRP), Left Ventricular Refractory Period (LVRP), Post Ventricular Atrial Refractory Period (PVARP), Right Ventricular Blanking Period (RVBP), Left Ventricular Blanking Period (LVBP), A-Blank after Right Ventricular Activity (ABaRV), and A-Blank after Left Ventricular Activity (ABaLV).

We introduce six new safety properties using invariants ($inv1$ - $inv6$). The first two safety properties state that during the refractory periods and blanking periods, the sensor and actuator must be *OFF*, and these sensor and actuator can be *ON* only after the refractory and blanking periods are elapsed. To check the refractory period after pacing or sensing activity, we need to store the time of pacing or sensing activity of ventricular chambers. We use a variable $PSRecord$ for this. The next four safety properties state that sensing and pacing of both ventricular chambers can occur only after blanking periods and refractory periods. Here we do not need an additional variable because after pacing or sensing in the atrial chamber, the clock resets.

$$
\begin{aligned}
&inv1 : PM\_Actuator\_A = ON \Rightarrow now \geq PSRecord + PVARP \wedge \\
&\qquad now \geq PSRecord + RVRP \wedge now \geq PSRecord + LVRP \wedge \\
&\qquad now \geq PSRecord + ABaLV \wedge now \geq PSRecord + ABaRV \\
&inv2 : PM\_Sensor\_A = ON \Rightarrow now \geq PSRecord + PVARP \wedge \\
&\qquad now \geq PSRecord + RVRP \wedge now \geq PSRecord + LVRP \\
&\qquad \wedge now \geq PSRecord + ABaLV \wedge now \geq PSRecord + ABaRV \\
&inv3 : PM\_Actuator\_RV = ON \Rightarrow now \geq ARP \wedge now \geq RVBP \\
&inv4 : PM\_Actuator\_LV = ON \Rightarrow now \geq ARP \wedge now \geq LVBP \\
&inv5 : PM\_Sensor\_RV = ON \Rightarrow now \geq ARP \wedge now \geq RVBP \\
&inv6 : PM\_Sensor\_LV = ON \Rightarrow now \geq ARP \wedge now \geq LVBP
\end{aligned}
$$

In this refinement we introduce a new guard in the event *tic* to formalize the guard conditions that progress the current time (*now*), and to model the desired system behaviours of sensing and pacing activities correctly. The provided guard synchronizes all pacing and sensing activities by considering intrinsic activities of pacing and sensing according to the BiSP. We do not have space to show the complete timing requirements as the guard conditions of *tic* event, but as an example we present that portion that shows the progress of *tic* corresponding to Fig. 3.

EVENT tic
WHEN
grd1: $(now < AVI \wedge PM\_Sensor\_LV = OFF \wedge PM\_Sensor\_RV = OFF \wedge$
$No\_Pace\_LV\_RV = 1)$
$\vee$
$(now \geq AVI \wedge now < PSRecord + PVARP \wedge PM\_Sensor\_LV = OFF \wedge$
$PM\_Sensor\_RV = OFF \wedge No\_Pace\_LV\_RV = 1 \wedge$
$(Pace\_RV = 2 \vee Thr\_State\_RV = TRUE))$
$\vee$
$(now \geq PSRecord + PVARP \wedge now < PSRecord + VAI \wedge PM\_Sensor\_LV = OFF \wedge$
$PM\_Sensor\_RV = OFF \wedge No\_Pace\_LV\_RV = 1 \wedge PM\_Sensor\_A = ON)$
$\vee$
...
...
THEN
act1 : $now := now + 1$
END

4.4 Model Validation and Analysis

In this section, we present the proof statistics by presenting detailed information about generated proof obligations, and validity of the models using ProB [6]. Validation refers to gaining confidence that the developed models are consistent with requirements.

Table 1. Proof Statistics

Model	Total number of POs	Automatic Proof	Interactive Proof
Abstract Model	0	0(0%)	0(0%)
First Refinement	102	101(99%)	1(1%)
Second Refinement	23	23(100%)	0(0%)
Third Refinement	59	57(98%)	2(2%)
Total	184	181(98.37%)	3(1.63%)

Event-B supports two types of validation activities namely *consistency checking* and *model analysis*. Consistency checking shows that a list of events preserves the given invariants, and refinement checking makes sure that one machine is a valid refinement of previous machines. Model analysis is done with the help of ProB, which explores traces of Event-B models. The ProB tool supports *automated consistency checking* and *constraint-based checking*. ProB may help to find possible deadlocks or hidden properties that may not be expressed by generated proof obligations. In our work, ProB animation helps to identify the desired behaviour of the CRT pacemaker in different scenarios and validates the developed formal models. This tool assists us in finding potential problems, and to improve the guard predicates of events. The ProB model checker was able to animate all the possible machines from abstract to concrete level, and to prove the absence of errors (no counter example exist). It should be noted that ProB uses all the described safety properties during model checking process to report any violation of safety properties against the formalized system behaviour. Table 1 shows the proof statistics of the development in the RODIN tool. This development results in 184(100 %) proof obligations, in which 181(98.37 %) are proved automatically, and the remaining 3(1.63 %) are proved interactively using the Rodin prover. These proofs are quite simple, and can be achieved with the help of simplifying predicates.

5 Discussion

Stepwise refinement played an important role in our effort to develop a CRT pacemaker. Stepwise refinement in various forms has long been a suggested approach in the development of dependable systems, and was championed by Harlan Mills in his work on Box-Structures [10]. As mentioned earlier, refinement is a core concept in Event-B development. Of interest to practitioners is how we decide on what to introduce in each new refinement. There may be no universally 'correct' pattern to follow. However, building on experience with earlier development of pacemaker models we chose the order of: (1) Introduce all possible hardware elements at the abstract level like actuators and sensors; (2) Introduce a clock; (3) Include thresholds; and (4) Include refractory and blanking periods by introducing a guard on the clock.

We have designed our CRT pacemaker using a *correct by construction* approach. We described the system requirements using set theoretical notations abstractly, that can be further refined incrementally to reach a concrete level similar to code. Event-B has very good tool support that allows us to prove given properties (mostly) automatically. Other formal modelling tools like VDM, Z, Alloy can be used in place of the Event-B modelling language without considering refinement notions. However, Simulink and SLDV modelling techniques cannot be used in place of these formal modelling languages, because Simulink and SLDV do not allow us to build a system abstractly. The developed formal model contains all the possible ranges for each constant and variable used in describing the system behaviour, so the Event-B concrete model can be used for different implementation purposes. Therefore we can use the concrete Event-B models of the CRT pacemaker for implementing an actual system considering hardware requirements using Simulink and SLDV. The tool support of Simulink and SLDV can help guarantee a correct implementation.

As far as we know, there are no published formal system requirements for the CRT pacemaker, but several research publications and clinical books provide informal requirements and discussion on clinical practices. We used such informal descriptions as a basis for this work. We also identified a list of safety properties in the refinement process to verify the correctness of overall formalized system behaviour, including newly introduced features. These safety properties guarantee that all possible executions of the system are safe, if the generated proof obligations are successfully discharged – and if we our list of safety properties is correct and complete. We have considered only the main safety properties (see all invariants) related to actuators and sensors under timing requirements. We can introduce additional safety properties in further refinements which may or may not include new features.

6 Related Work

The pacemaker has been formalized by several researchers around the world. There is a distributed real-time model of the pacemaker formalized in VDM by Macedo et al. [11], in which they addressed selected behaviour of the pacemaker. Gomes et al. [12] formalized the sequential model of the pacemaker using Z, and then they used Perfect Developer tool to produce source code. In [13], authors used the dual chamber implantable pacemaker as a case study for modelling and verification of control algorithms for medical devices in UPPAAL. Tuan et al. [14] designed a real-time model of the pacemaker using CSP, and the developed model was verified by a model checker Process Analysis Toolkit (PAT) in order to verify system properties. A detailed formalization of the one- and two-electrode pacemaker was presented in [15,16], where the models were developed in an incremental way using refinements in the Event-B modelling language. In this work, the authors developed operating modes considering advanced features like *threshold*, *hysteresis* and *rate adaptive*. A closed-loop model of the pacemaker and heart was presented in [15,17]. Méry et al. used EB2ALL [15,18]

to generate executable source code from the pacemaker specification in multiple programming languages.

7 Conclusion and Future Challenges

The CRT pacemaker is a complex medical device for sensing and pacing in the heart chambers. An actuator delivers a brief electrical stimulus when there is an absence of intrinsic heart activity within a bounded time interval, and the sensing and pacing activities of each chamber are synchronized so that the heart functions as normally as possible. Timing requirements are crucial to building a correct and effective model for the CRT pacemaker. However, the timing requirements for this sensing and pacing are extremely complex to analyze manually. Our proposed refinement based formal development of the CRT pacemaker captures what we think are the essential requirements, and enables us to verify that the model complies with the primary safety properties we derived. In the future we intend to augment these safety properties.

This paper presents a formal development of a CRT pacemaker using incremental refinement. As far as we know, this is the first formal model of the CRT pacemaker to analyze that the functional behaviour complies with its safety requirements. We used the Event-B modelling language, together with its associated tools, to develop the proof-based formal model using a refinement technique. Our incremental development reflects not only the many facets of the problem, but also that there is a learning process involved in understanding the problem and its ultimate possible solutions.

This formal model (or rather, future refinements of it) can be used to help certify CRT or multisite pacing devices. Our goal is to integrate formal models of CRT pacemakers and the heart, to model the closed-loop system for verifying the desired behaviour under relevant safety properties, and be able to guarantee the correctness of the functional behaviour of the CRT. We also intend to implement a CRT algorithm for a hardware platform by generating source code using EB2ALL from the formal model.

This work not only delivers a formal model of the CRT pacemaker, it builds on previous Event-B development of the 2-electrode pacemaker as described in the PACEMAKER Challenge. This is important because as we use the same technology/methodology to develop related medical devices we can begin to see principles for how to use that methodology so that development is more efficient. In our case, one of the principles we are starting to understand better is how to use stepwise refinement (in the Event-B sense) to build the requirements model.

Finally, whatever system we use to model the behaviour, the safety and dependability of the device depends on the *correctness* of our requirements model. We believe that we can use Event-B tools to validate these requirements with the help of clinical specialists, and correct any mistakes we may have in the current model.

References

1. Carayon, P., Wood, K.E.: Patient safety. Inf. Knowl. Syst. Manage. **8**(1–4), 23–46 (2009)
2. Maisel, W.H., Moynahan, M., Zuckerman, B.D., Gross, T.P., Tovar, O.H., Tillman, D.B., Schultz, D.B.: Pacemaker and ICD generator malfunctions: Analysis of food and drug administration annual reports. JAMA **295**(16), 1901–1906 (2006)
3. Boston scientific: pacemaker system specification. Technical report (2007). http://www.cas.mcmaster.ca/sqrl/SQRLDocuments/PACEMAKER.pdf
4. Dagstuhl seminar 14062: The pacemaker challenge: developing certifiable medical devices (2014)
5. Abrial, J.R.: Modeling in Event-B: System and Software Engineering, 1st edn. Cambridge University Press, New York (2010)
6. Leuschel, M., Butler, M.: ProB: A model checker for B. In: Araki, K., Gnesi, S., Mandrioli, D. (eds.) FME 2003. LNCS, vol. 2805, pp. 855–874. Springer, Heidelberg (2003)
7. Barold, S.S., Stroobandt, R.X., Sinnaeve, A.F.: Cardiac Pacemakers Step by Step. Futura Publishing (2004). ISBN 1-4051-1647-1
8. Project RODIN: Rigorous open development environment for complex systems (2004). http://rodin-b-sharp.sourceforge.net/
9. Wang, P., Kramer, A., Mark Estes, N.A., Hayes, D.L.: Timing cycles for biventricular pacing. Pacing Clin. Electrophysiol. **25**, 62–75 (2002)
10. Mills, H.D.: Stepwise refinement and verification in box-structured systems. IEEE Comput. **21**(6), 23–36 (1988)
11. Macedo, H.D., Larsen, P.G., Fitzgerald, J.S.: Incremental development of a distributed real-time model of a cardiac pacing system using VDM. In: Cuellar, J., Sere, K. (eds.) FM 2008. LNCS, vol. 5014, pp. 181–197. Springer, Heidelberg (2008)
12. Gomes, A.O., Oliveira, M.V.M.: Formal Specification of a cardiac pacing system. In: Cavalcanti, A., Dams, D.R. (eds.) FM 2009. LNCS, vol. 5850, pp. 692–707. Springer, Heidelberg (2009)
13. Jiang, Z., Pajic, M., Moarref, S., Alur, R., Mangharam, R.: Modeling and verification of a dual chamber implantable pacemaker. In: Flanagan, C., König, B. (eds.) TACAS 2012. LNCS, vol. 7214, pp. 188–203. Springer, Heidelberg (2012)
14. Tuan, L.A., Zheng, M.C., Tho, Q.T.: Modeling and verification of safety critical systems: A case study on pacemaker. In: Secure System Integration and Reliability Improvement, june 2010, pp. 23–32 (2010)
15. Singh, N.K.: Using Event-B for Critical Device Software Systems. Springer GmbH, London (2013)
16. Méry, D., Singh, N.K.: Functional behavior of a cardiac pacing system. Int. J. Discrete Event Control Syst. **1**(2), 129–149 (2011)
17. Méry, D., Singh, N.K.: Formalization of Heart Models Based on the Conduction of Electrical Impulses and Cellular Automata. In: Liu, Z., Wassyng, A. (eds.) FHIES 2011. LNCS, vol. 7151, pp. 140–159. Springer, Heidelberg (2012)
18. Méry, D., Singh, N.K.: Automatic code generation from Event-B models. In: Proceedings of the Second Symposium on Information and Communication Technology, SoICT 2011, pp. 179–188. ACM, New York (2011)